Docket No. 241737US2

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF: Atsushi MASUDA GAU:

SERIAL NO: New Application EXAMINER:

FILED: Herewith

FOR: DESIGN METHOD OF LOGIC CIRCUIT

REQUEST FOR PRIORITY

COMMISSIONER FOR PATENTS
ALEXANDRIA, VIRGINIA 22313

SIR:

☐ Full benefit of the filing date of U.S. Application Serial Number , filed , is claimed pursuant to the provisions of **35 U.S.C. §120**.

☐ Full benefit of the filing date(s) of U.S. Provisional Application(s) is claimed pursuant to the provisions of **35 U.S.C. §119(e)**: **Application No.** **Date Filed**

■ Applicants claim any right to priority from any earlier filed applications to which they may be entitled pursuant to the provisions of **35 U.S.C. §119**, as noted below.

In the matter of the above-identified application for patent, notice is hereby given that the applicants claim as priority:

COUNTRY	APPLICATION NUMBER	MONTH/DAY/YEAR
Japan	2002-354208	December 5, 2002

Certified copies of the corresponding Convention Application(s)

■ are submitted herewith

☐ will be submitted prior to payment of the Final Fee

☐ were filed in prior application Serial No. filed

☐ were submitted to the International Bureau in PCT Application Number
Receipt of the certified copies by the International Bureau in a timely manner under PCT Rule 17.1(a) has been acknowledged as evidenced by the attached PCT/IB/304.

☐ (A) Application Serial No.(s) were filed in prior application Serial No. filed ; and

☐ (B) Application Serial No.(s)

- ☐ are submitted herewith
- ☐ will be submitted prior to payment of the Final Fee

Respectfully Submitted,

OBLON, SPIVAK, McCLELLAND,
MAIER & NEUSTADT, P.C.

Marvin J. Spivak
Registration No. 24,913

C. Irvin McClelland
Registration Number 21,124

22850

Tel. (703) 413-3000
Fax. (703) 413-2220
(OSMMN 05/03)

日　本　国　特　許　庁
JAPAN　PATENT　OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application:　２００２年１２月　５日

出願番号
Application Number:　特願２００２－３５４２０８

[ST.10/C]:　[ＪＰ２００２－３５４２０８]

出願人
Applicant(s):　株式会社東芝

２００２年１２月２７日

特許庁長官
Commissioner,
Japan Patent Office　太田信一郎

出証番号　出証特２００２－３１０２６１４

【書類名】　特許願
【整理番号】　ASB023110
【提出日】　平成14年12月 5日
【あて先】　特許庁長官殿
【国際特許分類】　G06F 17/50
【発明の名称】　論理回路設計方法、論理回路設計プログラムおよび論理回路設計装置
【請求項の数】　15
【発明者】
【住所又は居所】　神奈川県川崎市幸区小向東芝町１番地　株式会社東芝マイクロエレクトロニクスセンター内
【氏名】　増田　篤司
【特許出願人】
【識別番号】　000003078
【氏名又は名称】　株式会社　東芝
【代理人】
【識別番号】　100083806
【弁理士】
【氏名又は名称】　三好　秀和
【電話番号】　03-3504-3075
【選任した代理人】
【識別番号】　100068342
【弁理士】
【氏名又は名称】　三好　保男
【選任した代理人】
【識別番号】　100100712
【弁理士】
【氏名又は名称】　岩▲崎▼　幸邦

【選任した代理人】

【識別番号】　100100929

【弁理士】

【氏名又は名称】　川又　澄雄

【選任した代理人】

【識別番号】　100108707

【弁理士】

【氏名又は名称】　中村　友之

【選任した代理人】

【識別番号】　100095500

【弁理士】

【氏名又は名称】　伊藤　正和

【選任した代理人】

【識別番号】　100101247

【弁理士】

【氏名又は名称】　高橋　俊一

【選任した代理人】

【識別番号】　100098327

【弁理士】

【氏名又は名称】　高松　俊雄

【手数料の表示】

【予納台帳番号】　001982

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　論理回路設計方法、論理回路設計プログラムおよび論理回路設計装置

【特許請求の範囲】

【請求項１】　論理回路のアルゴリズムを、複数の演算子を有する動作記述から、前記演算子を実行する演算ノードを実行順に配列したデータフロー図に変換するステップと、

前記データフロー図に実行ステップを割り当て、前記演算ノードからの出力データを記憶するレジスタを前記実行ステップの実行後に挿入するステップと、

前記演算ノードとして機能する演算器と前記レジスタとして機能する記憶素子を有する前記論理回路のデータパスと、前記データパスの制御情報を生成するステップと、

前記演算子を実行する前記演算器を前記演算子から検索可能であり、前記記憶素子の機能をする前記レジスタが記憶するデータを出力する前記演算子を前記実行ステップと前記記憶素子から検索可能である演算子演算器データベースを生成するステップとを有することを特徴とする論理回路設計方法。

【請求項２】　前記動作記述と前記データフロー図に基づいて、前記演算ノードで実行する前記演算子を前記演算ノードから検索可能であり、前記レジスタが記憶するデータを出力する前記演算ノードで実行する前記演算子を前記レジスタから検索可能であるノード演算子データベースを生成するステップと、

前記データフロー図と前記データパスに基づいて、前記演算ノードとして機能する前記演算器を前記演算ノードから検索可能であり、前記記憶素子の機能をする前記レジスタを前記実行ステップと前記記憶素子から検索可能であるノード演算器データベースを生成するステップをさらに有し、

前記演算子演算器データベースを生成するステップが、前記ノード演算子データベースと前記ノード演算器データベースに基づくこととを特徴とする請求項１に記載の論理回路設計方法。

【請求項３】　前記データパスと前記制御情報に基づいて、前記論理回路のレジスタ転送レベル記述を生成するステップと、

前記動作記述に入力データを代入し、前記演算子からの出力データを計算するステップと、

前記レジスタ転送レベル記述に前記入力データを代入し、前記実行ステップで前記記憶素子が記憶する記憶データを計算するステップと、

演算子演算器データベースに基づいて、計算された前記記憶データの前記実行ステップと前記記憶素子から前記演算子を検索するステップと、

検索された前記演算子の前記出力データと、前記記憶データの異同を判定するステップとをさらに有することを特徴とする請求項１または請求項２に記載の論理回路設計方法。

【請求項４】　演算子演算器データベースに基づいて、検索された前記演算子から前記演算器を検索するステップをさらに有することを特徴とする請求項１乃至請求項３のいずれか１つに記載の論理回路設計方法。

【請求項５】　予想される全実行時間が最短になるようにデータフロー図を変更するステップとをさらに有することを特徴とする請求項１乃至請求項４のいずれか１つに記載の論理回路設計方法。

【請求項６】　論理回路のアルゴリズムを、複数の演算子を有する動作記述から、前記演算子を実行する演算ノードを実行順に配列したデータフロー図に変換する手順と、

前記データフロー図に実行ステップを割り当て、前記演算ノードからの出力データを記憶するレジスタを前記実行ステップの実行後に挿入する手順と、

前記演算ノードとして機能する演算器と前記レジスタとして機能する記憶素子を有する前記論理回路のデータパスと、前記データパスの制御情報を生成する手順と、

前記演算子を実行する前記演算器を前記演算子から検索可能であり、前記記憶素子の機能をする前記レジスタが記憶するデータを出力する前記演算子を前記実行ステップと前記記憶素子から検索可能である演算子演算器データベースを生成する手順をコンピュータに実行させるための論理回路設計プログラム。

【請求項７】　前記動作記述と前記データフロー図に基づいて、前記演算ノードで実行する前記演算子を前記演算ノードから検索可能であり、前記レジスタが

記憶するデータを出力する前記演算ノードで実行する前記演算子を前記レジスタから検索可能であるノード演算子データベースを生成する手順と、

前記データフロー図と前記データパスに基づいて、前記演算ノードとして機能する前記演算器を前記演算ノードから検索可能であり、前記記憶素子の機能をする前記レジスタを前記実行ステップと前記記憶素子から検索可能であるノード演算器データベースを生成する手順をさらにコンピュータに実行させるためにあり、

前記演算子演算器データベースを生成する手順が、前記ノード演算子データベースと前記ノード演算器データベースに基づくこととを特徴とする請求項６に記載の論理回路設計プログラム。

【請求項８】　前記データパスと前記制御情報に基づいて、前記論理回路のレジスタ転送レベル記述を生成する手順と、

前記動作記述に入力データを代入し、前記演算子からの出力データを計算する手順と、

前記レジスタ転送レベル記述に前記入力データを代入し、前記実行ステップで前記記憶素子が記憶する記憶データを計算する手順と、

演算子演算器データベースに基づいて、計算された前記記憶データの前記実行ステップと前記記憶素子から前記演算子を検索する手順と、

検索された前記演算子の前記出力データと、前記記憶データの異同を判定する手順をさらにコンピュータに実行させるための請求項６または請求項７に記載の論理回路設計プログラム。

【請求項９】　演算子演算器データベースに基づいて、検索された前記演算子から前記演算器を検索する手順をさらにコンピュータに実行させるための請求項６乃至請求項８のいずれか１つに記載の論理回路設計プログラム。

【請求項１０】　予想される全実行時間が最短になるようにデータフロー図を変更する手順をさらにコンピュータに実行させるための請求項６乃至請求項９のいずれか１つに記載の論理回路設計プログラム。

【請求項１１】　論理回路のアルゴリズムを、複数の演算子を有する動作記述から、前記演算子を実行する演算ノードを実行順に配列したデータフロー図に変

換する構文解析部と、

前記データフロー図に実行ステップを割り当て、前記演算ノードからの出力データを記憶するレジスタを前記実行ステップの実行後に挿入するスケジューリング部と、

前記演算ノードとして機能する演算器と前記レジスタとして機能する記憶素子を有する前記論理回路のデータパスと、前記データパスの制御情報を生成するハードウェア割付部と、

前記演算子を実行する前記演算器を前記演算子から検索可能であり、前記記憶素子の機能をする前記レジスタが記憶するデータを出力する前記演算子を前記実行ステップと前記記憶素子から検索可能である演算子演算器データベースを生成する演算子演算器対応情報解析部を有することを特徴とする論理回路設計装置。

【請求項１２】　前記動作記述と前記データフロー図に基づいて、前記演算ノードで実行する前記演算子を前記演算ノードから検索可能であり、前記レジスタが記憶するデータを出力する前記演算ノードで実行する前記演算子を前記レジスタから検索可能であるノード演算子データベースを生成するノード演算子対応情報解析部と、

前記データフロー図と前記データパスに基づいて、前記演算ノードとして機能する前記演算器を前記演算ノードから検索可能であり、前記記憶素子の機能をする前記レジスタを前記実行ステップと前記記憶素子から検索可能であるノード演算器データベースを生成するノード演算器対応情報解析部をさらに有し、

前記演算子演算器データベースを生成することが、前記ノード演算子データベースと前記ノード演算器データベースに基づくこととを特徴とする請求項１１に記載の論理回路設計装置。

【請求項１３】　前記データパスと前記制御情報に基づいて、前記論理回路のレジスタ転送レベル記述を生成するレジスタ転送レベル記述生成部と、

前記動作記述に入力データを代入し、前記演算子からの出力データを計算する動作記述シミュレーション部と、

前記レジスタ転送レベル記述に前記入力データを代入し、前記実行ステップで前記記憶素子が記憶する記憶データを計算するレジスタ転送レベルシミュレーシ

ョン部と、

演算子演算器データベースに基づいて、計算された前記記憶データの前記実行ステップと前記記憶素子から前記演算子を検索する演算子検索部と、

検索された前記演算子の前記出力データと、前記記憶データの異同を判定する記憶データ判定部をさらに有することを特徴とする請求項１１または請求項１２に記載の論理回路設計装置。

【請求項１４】　演算子演算器データベースに基づいて、検索された前記演算子から前記演算器を検索する演算器検索部をさらに有することを特徴とする請求項１１乃至請求項１３のいずれか１つに記載の論理回路設計装置。

【請求項１５】　予想される全実行時間が最短になるようにデータフロー図を変更する最適化部をさらに有することを特徴とする請求項１１乃至請求項１４のいずれか１つに記載の論理回路設計装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、レジスタ転送レベル（ＲＴＬ）記述を自動生成する高位合成技術に関し、特に、動作記述からＲＴＬ記述に変換する論理回路設計方法に関する。

【０００２】

【従来の技術】

従来、論理回路設計では、動作記述の中間信号と論理回路の演算器との対応表が出力された。（例えば、特許文献１参照。）。

【０００３】

しかし、さらに高位合成されたＲＴＬ記述では、実行ステップ毎にデータが記憶素子に記憶され、１つの演算器が繰り返し演算を行う。このため、対応表は存在せず、実行ステップ毎の対応表も存在しなかった。ＲＴＬ記述の論理回路のシミュレーション結果に不具合があった場合は、設計者がRTL記述のみを用いてそのシミュレーション結果を解析した。RTL記述は、動作時間や条件などにより動作する部分が変わるので、この解析には非常に多くの時間を要した。

【０００４】

【特許文献１】

特開平11-73447号公報（請求項１、図６）

【０００５】

【発明が解決しようとする課題】

本発明は、上記事情に鑑みてなされたものであり、その目的とするところは、動作記述から変換されたＲＴＬ記述での論理回路のシミュレーション結果を迅速に解析するための論理回路設計方法を提供することにある。

【０００６】

本発明の目的は、コンピュータが動作記述から変換されたＲＴＬ記述での論理回路のシミュレーション結果を迅速に解析するための論理回路設計プログラムを提供することにある。

【０００７】

本発明の目的は、動作記述から変換されたＲＴＬ記述での論理回路のシミュレーション結果を迅速に解析するための論理回路設計装置を提供することにある。

【０００８】

【課題を解決するための手段】

上記問題点を解決するための本発明の第１の特徴は、論理回路のアルゴリズムを、複数の演算子を有する動作記述から、演算子を実行する演算ノードを実行順に配列したデータフロー図に変換することと、

データフロー図に実行ステップを割り当て、演算ノードからの出力データを記憶するレジスタを実行ステップの実行後に挿入することと、

演算ノードとして機能する演算器とレジスタとして機能する記憶素子を有する論理回路のデータパスと、データパスの制御情報を生成することと、

演算子を実行する演算器を演算子から検索可能であり、記憶素子の機能をするレジスタが記憶するデータを出力する演算子を実行ステップと記憶素子から検索可能である演算子演算器データベースを生成することとを有する論理回路設計方法にある。

【０００９】

本発明の第２の特徴は、論理回路のアルゴリズムを、複数の演算子を有する動

作記述から、演算子を実行する演算ノードを実行順に配列したデータフロー図に変換する手順と、

データフロー図に実行ステップを割り当て、演算ノードからの出力データを記憶するレジスタを実行ステップの実行後に挿入する手順と、

演算ノードとして機能する演算器とレジスタとして機能する記憶素子を有する論理回路のデータパスと、データパスの制御情報を生成する手順と、

演算子を実行する演算器を演算子から検索可能であり、記憶素子の機能をするレジスタが記憶するデータを出力する演算子を実行ステップと記憶素子から検索可能である演算子演算器データベースを生成する手順をコンピュータに実行させるための論理回路設計プログラムにある。

【００１０】

本発明の第３の特徴は、論理回路のアルゴリズムを、複数の演算子を有する動作記述から、演算子を実行する演算ノードを実行順に配列したデータフロー図に変換する構文解析部と、

データフロー図に実行ステップを割り当て、演算ノードからの出力データを記憶するレジスタを実行ステップの実行後に挿入するスケジューリング部と、

演算ノードとして機能する演算器とレジスタとして機能する記憶素子を有する論理回路のデータパスと、データパスの制御情報を生成するハードウェア割付部と、

演算子を実行する演算器を演算子から検索可能であり、記憶素子の機能をするレジスタが記憶するデータを出力する演算子を実行ステップと記憶素子から検索可能である演算子演算器データベースを生成する演算子演算器対応情報解析部を有する論理回路設計装置にある。

【００１１】

【発明の実施の形態】

次に、図面を参照して、本発明の実施の形態について説明する。以下の図面の記載において、同一又は類似の部分には同一又は類似の符号を付している。また、図面は模式的なものであり、現実のものとは異なることに留意すべきである。

【００１２】

（論理回路設計装置）

本発明の実施の形態に係る論理回路設計装置１は、図１に示すように、構文解析部２、スケジューリング部３、最適化部４、ハードウェア割付部５、ＲＴＬ記述生成部６、ノード演算子対応情報解析部７、ノード演算器対応情報解析部８、演算子演算器対応情報解析部９、動作記述シミュレーション部１０、ＲＴＬ記述シミュレーション部１１、シミュレーション結果比較部１２と入出力部１６を有している。シミュレーション結果比較部１２は、演算子検索部１３、記憶データ判定部１４と演算器検索部１５を有している。

【００１３】

構文解析部２は、図２に示すように、論理回路のアルゴリズムを、複数の演算子を有する動作記述Ｄ１を入力し、演算子を実行する演算ノードを実行順に配列した未加工データフロー図（ＤＦＧ）Ｄ２を出力する。

【００１４】

スケジューリング部３は、未加工データフロー図Ｄ２を入力し、未加工データフロー図Ｄ２に実行ステップを割り当てる。スケジューリング部３は、演算ノードからの出力データを記憶するレジスタを、未加工データフロー図Ｄ２の実行ステップの実行後に挿入する。スケジューリング部３は、スケジューリングデータフロー図Ｄ３を出力する。

【００１５】

最適化部４は、スケジューリングデータフロー図Ｄ３を入力する。最適化部４は、論理回路のアルゴリズムの予想される全実行時間が最短になるように、スケジューリングデータフロー図Ｄ３を変更し、最適化データフロー図Ｄ４を出力する。

【００１６】

ハードウェア割付部５は、最適化データフロー図Ｄ４を入力する。ハードウェア部５は、演算ノードとして機能する演算器とレジスタとして機能する記憶素子を有する論理回路のデータパスＤ５と、データパスの制御情報Ｄ５を生成する。

【００１７】

ＲＴＬ記述生成部６は、データパスと制御情報Ｄ５を入力する。ＲＴＬ記述生

成部６は、データパスと制御情報Ｄ５に基づいて、論理回路のＲＴＬ記述Ｄ６を生成する。ＲＴＬ記述生成部６は、ＲＴＬ記述Ｄ６を出力する。

【００１８】

ノード演算子対応情報解析部７は、動作記述Ｄ１、スケジューリングデータフロー図Ｄ３と最適化データフロー図Ｄ４を入力する。ノード演算子対応情報解析部７は、動作記述Ｄ１、スケジューリングデータフロー図Ｄ３と最適化データフロー図Ｄ４に基づいて、ノード演算子データベースＤ７を生成する。ノード演算子データベースＤ７は、演算ノードで実行する演算子を演算ノードから検索可能であり、レジスタが記憶するデータを出力する演算ノードで実行する演算子をレジスタから検索可能である。

【００１９】

ノード演算器対応情報解析部８は、最適化データフロー図Ｄ４とデータパスＤ５を入力する。ノード演算器対応情報解析部８は、最適化データフロー図Ｄ４とデータパスＤ５に基づいて、ノード演算器データベースＤ８を生成する。ノード演算器データベースＤ８は、演算ノードとして機能する演算器を演算ノードから検索可能であり、記憶素子の機能をするレジスタを実行ステップと記憶素子から検索可能である。

【００２０】

演算子演算器対応情報解析部９は、ノード演算子データベースＤ７とノード演算器データベースＤ８を入力する。演算子演算器対応情報解析部９は、ノード演算子データベースＤ７とノード演算器データベースＤ８に基づいて、演算子演算器データベースＤ９を生成する。演算子演算器データベースＤ９は、演算子を実行する演算器を演算子から検索可能であり、記憶素子の機能をするレジスタが記憶するデータを出力する演算子を実行ステップと記憶素子から検索可能である。

【００２１】

動作記述シミュレーション部１０は、動作記述Ｄ１と入力データＤ１３を入力する。動作記述シミュレーション部１０は、動作記述Ｄ１に入力データＤ１３を代入し、演算子からの出力データＤ１０を計算する。

【００２２】

ＲＴＬ記述シミュレーション部１１は、ＲＴＬ記述Ｄ６と入力データＤ１３を入力する。ＲＴＬ記述シミュレーション部１１は、RTL記述Ｄ６に入力データＤ１３を代入し、実行ステップ毎に記憶素子が記憶する記憶データＤ１１を計算する。

【００２３】

シミュレーション結果比較部１２の演算子検索部１３は、演算子演算器データベースＤ９と、記憶データＤ１１を記憶した際の実行ステップと記憶素子を入力する。演算子検索部１３は、演算子演算器データベースＤ９に基づいて、計算された記憶データの実行ステップと記憶素子から演算子を検索する。

【００２４】

シミュレーション結果比較部１２の記憶データ判定部１４は、記憶データＤ１１と、検索された演算子の出力データを入力する。記憶データ判定部１４は、検索された演算子の出力データと記憶データの異同を判定する。検索された演算子の出力データと記憶データが異なる場合は、ＲＴＬ記述Ｄ６に不具合があると判定する。この判定に基づいて、ＲＴＬ記述Ｄ６の不具合の有無Ｄ１２を出力する。ＲＴＬ記述Ｄ６に不具合がある場合は、さらに、記憶データＤ１１を記憶した際の実行ステップ、記憶素子と演算子Ｄ１２を出力する。

【００２５】

シミュレーション結果比較部１２の演算器検索部１５は、検索された演算子の出力データと記憶データが異なる場合に、演算子演算器データベースＤ９と検索された演算子を入力する。演算器検索部１５は、演算子演算器データベースＤ９に基づいて、検索された演算子から演算器を検索する。演算器検索部１５は、検索された演算器Ｄ１２を出力する。

【００２６】

入出力部１６は、動作記述Ｄ１と入力データＤ１３を入力する。入出力部１６は、ＲＴＬ記述Ｄ６とＲＴＬ記述Ｄ６の不具合の有無、不具合の発生した実行ステップ、記憶素子、演算子と演算器Ｄ１２を出力する。

【００２７】

高位合成でC記述等の動作記述Ｄ１からRTL記述Ｄ６を生成する際に、動作記述

Ｄ１とRTL記述Ｄ６の両者を対応づける演算子演算器データベースＤ９を生成する。RTL記述Ｄ６の検証に、動作記述Ｄ１の検証に用いたテストベクトルを使うことができ、かつ、動作記述Ｄ１との比較検証をすることにより、不具合が生じた場合には、動作記述Ｄ１、RTL記述Ｄ６の両者での不具合箇所が特定できる。このことにより、動作記述Ｄ１やRTL記述Ｄ６の修正が容易に行えるようになり、設計、検証、不具合修正にかかる時間を大幅に短縮することができる。

【００２８】

（論理回路設計方法）

本発明の実施の形態に係る論理回路設計方法は、図３に示すように、ステップＳ１において、構文解析部２で、構文解析をする。論理回路のアルゴリズムを、複数の演算子を有する動作記述Ｄ１から、演算子を実行する演算ノードを実行順に配列した未加工データフロー図（ＤＦＧ）Ｄ２に変換する。

【００２９】

ステップＳ２において、スケジューリング部３で、スケジューリングをする。未加工データフロー図Ｄ２に実行ステップを割り当て、演算ノードにどの実行ステップで実行するかという時間情報が添付される。未加工データフロー図Ｄ２に、演算ノードからの出力データを記憶するレジスタを実行ステップの実行後に挿入する。このレジスタにより、実行ステップをまたがるデータを保持することができる。スケジューリング部３は、このように加工された未加工データフロー図Ｄ２をスケジューリングデータフロー図Ｄ３として出力する。

【００３０】

ステップＳ３において、最適化部４で、予想される全実行時間が最短になるようにスケジューリングデータフロー図Ｄ３を変更する。このように変更されたスケジューリングデータフロー図Ｄ３を最適化データフロー図Ｄ４として出力する。

【００３１】

ステップＳ４において、ハードウェア割付部５で、演算ノードとして機能する演算器とレジスタとして機能する記憶素子を有する論理回路のデータパスとこのデータパスの制御情報Ｄ５を生成する。

【００３２】

ステップＳ５において、ＲＴＬ記述生成部６で、データパスと制御情報Ｄ５に基づいて、ＲＴＬ記述Ｄ６を生成する。

【００３３】

ステップＳ６において、ＲＴＬ記述シミュレーション部１１で、実行ステップｉに、最初に実行される実行ステップ１を設定する。

【００３４】

ステップＳ７において、ＲＴＬ記述シミュレーション部１１で、RTL記述Ｄ６のシミュレーションを行う。RTL記述Ｄ６に入力データＤ１３を代入し、実行ステップｉで演算器が出力するデータと記憶素子が記憶する記憶データＤ１１を計算する。

【００３５】

一方、ステップＳ２とＳ３の実行後に、ステップＳ８において、ノード演算子対応情報解析部７で、動作記述Ｄ１とスケジューリングデータフロー図Ｄ３と最適化データフロー図Ｄ４に基づいて、ノード演算子データベースＤ７を生成する。

【００３６】

ステップＳ４の実行後に、ステップＳ９において、ノード演算器対応情報解析部８で、最適化データフロー図Ｄ４とデータパスＤ５に基づいて、ノード演算器データベースＤ８を生成する。

【００３７】

ステップＳ１０において、演算子演算器対応情報解析部９で、ノード演算子データベースＤ７とノード演算器データベースＤ８に基づいて、演算子演算器データベースＤ９を生成する。

【００３８】

ステップＳ１１において、動作記述シミュレーション部１０で、動作記述Ｄ１のシミュレーションを行う。動作記述Ｄ１に入力データＤ１３を代入し、演算子からの出力データＤ１０を計算する。動作記述を実際にコンパイルし実行した場合の各演算子が出力する中間データを計算する。

【００３９】

ステップＳ７とＳ１１の実行後に、ステップＳ１２において、シミュレーション結果比較部１２の演算子検索部１３で、演算子演算器データベースＤ９に基づいて、計算された記憶データＤ１１の実行ステップｉと記憶素子から演算子を検索する。

【００４０】

ステップＳ１３において、記憶データ判定部１４で、検索された演算子の出力データＤ１０と、記憶データＤ１１の異同を判定する。異なるとの判定の場合は、ステップＳ１６に進み、同じとの判定の場合は、ステップＳ１４に進む。

【００４１】

ステップＳ１６において、演算器検索部１５で、演算子演算器データベースＤ９に基づいて、検索された演算子から演算器を検索する。

【００４２】

ステップＳ１７において、論理回路設計装置１あるいはそのオペレータが、検索された演算器に基づいて、ＲＴＬ記述Ｄ６をデバックする。ステップＳ５に戻る。なお、戻る先は、ステップＳ５に限らない。再度シミュレーションされるＲＴＬ記述がデバックされたＲＴＬ記述であれば、ステップＳ６に戻ってもよい。

【００４３】

一方、ステップＳ１４において、シミュレーション結果比較部１２で、実行ステップｉが最適化データフロー図Ｄ４の最大の実行ステップ以上であるか否かを判定する。実行ステップｉが最大の実行ステップ以上でない場合は、ステップＳ１５に進む。ステップＳ１５において、実行ステップｉを１つ大きくする。そして、ステップＳ７に進む。実行ステップｉが最大の実行ステップ以上である場合は、論理回路設計方法をストップする。このように、動作記述Ｄ１のミュレーション結果である演算子の出力データＤ１０と、RTL記述Ｄ６のシミュレーション結果である記憶素子の記憶データＤ１１を、実行ステップ毎に比較することで、動作記述と高位合成されたRTL記述との比較検証が行われる。

【００４４】

動作記述Ｄ１とRTL記述の比較検証をすることにより、不具合が生じた場合に

は、動作記述Ｄ１、RTL記述Ｄ６の両者での不具合箇所が特定できる。このことにより、動作記述Ｄ１やRTL記述Ｄ６の修正が容易に行えるようになり、設計、検証、不具合修正にかかる時間を大幅に短縮することができる。

【００４５】

尚、RTL記述Ｄ６のシミュレーションでは、1実行ステップ毎にデータを比較しつつシミュレーションを行っている。これに限らず、RTL記述Ｄ６の全実行ステップの実行後に、動作記述Ｄ１とRTL記述の比較検証を行ってもよい。ただ、先の実行ステップでの不具合が、後の実行ステップでの不具合を起こす場合がある一方で、先の実行ステップでの不具合のデバックが、後の実行ステップで不具合を起こす場合もある。従って、動作記述Ｄ１とRTL記述の比較検証の時期は、1実行ステップ毎か、全実行ステップの実行後かを、場合により使い分けることが望ましい。

【００４６】

（実施例１）

実施例１では、論理回路設計装置１で論理回路設計方法を用いた。実施例１の論理回路設計方法は、図３のステップＳ１において、図４に示すような複数の演算子ｏｐ１乃至ｏｐ４を有する動作記述Ｄ１から、図５に示すような演算子ｏｐ１乃至ｏｐ４を実行する演算ノードＮ１乃至Ｎ４を実行順に配列した未加工データフロー図Ｄ２に変換する。演算子ｏｐ１乃至ｏｐ４は、実際には内部のデータとして管理され、識別番号やポインタ等で認識される。入力データＤ１３の引数ｉ１とｉ２で、演算ノードＮ１において、和算の演算子ｏｐ１を実行する。引数ｉ１とｉ２で、演算ノードＮ２において、引き算の演算子ｏｐ３を実行する。演算ノードＮ１の演算子ｏｐ１の出力と演算ノードＮ２の演算子ｏｐ２の出力で、演算ノードＮ３において、積算の演算子ｏｐ２を実行する。演算ノードＮ３の演算子ｏｐ２の出力と定数１で、演算ノードＮ４において、和算の演算子ｏｐ４を実行する。

【００４７】

ステップＳ２において、図６に示すように、図５の未加工データフロー図Ｄ２に実行ステップ１乃至３を割り当てる。演算ノードＮ１とＮ２は、実行ステップ

１で実行するという時間情報が添付される。演算ノードＮ３は、実行ステップ２で実行するという時間情報が添付される。演算ノードＮ４は、実行ステップ３で実行するという時間情報が添付される。未加工データフロー図Ｄ２に、演算ノードＮ１乃至Ｎ４からの出力データを記憶するレジスタＲ１乃至Ｒ４を実行ステップ１乃至３の実行後に挿入する。スケジューリング部３は、このように加工した図５の未加工データフロー図Ｄ２を、図６のスケジューリングデータフロー図Ｄ３として出力する。

【００４８】

ステップＳ３において、スケジューリングデータフロー図Ｄ３を最適化するが、既に、予想される全実行時間が最短であるので、出力された最適化データフロー図Ｄ４は、スケジューリングデータフロー図Ｄ３に等しい。

【００４９】

ステップＳ８において、図４の動作記述Ｄ１と図６のスケジューリングデータフロー図Ｄ３に基づいて、図７に示すようなノード演算子データベースＤ７を生成する。ノード演算子データベースＤ７は、複数のノード演算子レコード２１を有している。ノード演算子レコード２１は、ノードフィールド２２と演算子フィールド２３を有している。ノードフィールド２２では、演算ノードＮ１乃至Ｎ４とレジスタＲ１乃至Ｒ４が記憶される。演算子フィールド２３では、演算子ｏｐ１乃至ｏｐ４が記憶される。演算子ｏｐ１乃至ｏｐ４とその演算子ｏｐ１乃至ｏｐ４を実行する演算ノードＮ１乃至Ｎ４が、同じノード演算子レコード２１内に記憶される。また、演算子ｏｐ１乃至ｏｐ４とその演算子ｏｐ１乃至ｏｐ４の出力データを記憶するレジスタＲ１乃至Ｒ４が、同じノード演算子レコード２１内に記憶される。演算ノードＮ１乃至Ｎ４又はレジスタＲ１乃至Ｒ４から、同じノード演算子レコード２１内に記憶された演算子ｏｐ１乃至ｏｐ４を検索することが可能である。

【００５０】

ステップＳ４において、ハードウェア割付部５で、図８に示すような論理回路のデータパスＤ５と、このデータパスの制御情報Ｄ５を生成する。データパスＤ５は、演算ノードＮ１乃至Ｎ４として機能する演算器Ａ１乃至Ａ３と、レジスタ

Ｒ１乃至Ｒ４として機能する記憶素子Ｍ１とＭ２と、制御情報Ｄ５に基づいて伝達するデータを切り換えるセレクターＣ１乃至Ｃ３を有する。

【００５１】

ステップＳ９において、図６のスケジューリングデータフロー図Ｄ３と図８のデータパスＤ５に基づいて、図９に示すようなノード演算器データベースＤ８を生成する。ノード演算器データベースＤ８は、複数のノード演算器レコード２４を有している。ノード演算器レコード２４は、演算器フィールド２５、ノードフィールド２６と実行ステップフィールド２７を有している。演算器フィールド２５では、演算器Ａ１乃至Ａ３と記憶素子Ｍ１、Ｍ２が記憶される。ノードフィールド２２では、演算ノードＮ１乃至Ｎ４とレジスタＲ１乃至Ｒ４が記憶される。実行ステップフィールド２７では、実行ステップ１乃至３が記憶される。演算器Ａ１乃至Ａ３と、その演算器Ａ１乃至Ａ３で実行する演算ノードＮ１乃至Ｎ４と、その演算ノードＮ１乃至Ｎ４が実行する実行ステップが、同じノード演算器レコード２４内に記憶される。また、記憶素子Ｍ１、Ｍ２と、その記憶素子Ｍ１、Ｍ２で記憶するレジスタＲ１乃至Ｒ４と、そのレジスタＲ１乃至Ｒ４が記憶する実行ステップが、同じノード演算器レコード２４内に記憶される。

【００５２】

すなわち、図６のスケジューリングデータフロー図Ｄ３と図８のデータパスＤ５を参照しながら、図９のノード演算器データベースＤ８を見ると、実行ステップ１において、演算器Ａ１が演算ノードＮ１として機能し、演算器Ａ２が演算ノードＮ２として機能し、演算器Ａ３は働かず、記憶素子Ｍ１がレジスタＲ１として機能し、記憶素子Ｍ２がレジスタＲ２として機能する。実行ステップ２において、演算器Ａ１と演算器Ａ２は働かず、演算器Ａ３が演算ノードＮ３として機能し、記憶素子Ｍ１がレジスタＲ３として機能し、記憶素子Ｍ２は働かない。実行ステップ３において、演算器Ａ１が演算ノードＮ４として機能し、演算器Ａ２と演算器Ａ３は働かず、記憶素子Ｍ１がレジスタＲ４として機能し、記憶素子Ｍ２は働かない。

【００５３】

図９のノード演算器データベースＤ８によれば、演算ノードＮ１乃至Ｎ４から

、同じノード演算器レコード２４内に記憶された演算器Ａ１乃至Ａ３を検索することが可能である。記憶素子Ｍ１、Ｍ２と実行ステップ１乃至３から同じノード演算器レコード２４内に記憶されたレジスタＲ１乃至Ｒ４を検索することが可能である。

【００５４】

ステップＳ１０において、図７のノード演算子データベースＤ７と図９のノード演算器データベースＤ８に基づいて、図１０に示すような演算子演算器データベースＤ９を生成する。演算子演算器データベースＤ９は、複数の演算子演算器レコード２８を有している。演算子演算器レコード２８は、演算器フィールド２９、演算子フィールド３０と実行ステップフィールド３１を有している。演算器フィールド２９では、演算器Ａ１乃至Ａ３と記憶素子Ｍ１、Ｍ２が記憶される。演算子フィールド３０では、演算子ｏｐ１乃至ｏｐ４が記憶される。実行ステップフィールド３１では、実行ステップ１乃至３が記憶される。演算器Ａ１乃至Ａ３と、その演算器Ａ１乃至Ａ３で実行する演算子ｏｐ１乃至ｏｐ４と、その演算子ｏｐ１乃至ｏｐ４が実行する実行ステップが、同じ演算子演算器レコード２８内に記憶される。また、記憶素子Ｍ１、Ｍ２と、その記憶素子Ｍ１、Ｍ２で記憶するデータを出力する演算子ｏｐ１乃至ｏｐ４と、その演算子ｏｐ１乃至ｏｐ４が実行する実行ステップが、同じ演算子演算器レコード２８内に記憶される。

【００５５】

すなわち、図７のノード演算子データベースＤ７と図９のノード演算器データベースＤ８を参照しながら、図１０の演算子演算器データベースＤ９を見ると、実行ステップ１において、演算器Ａ１が演算子ｏｐ１として機能し、演算器Ａ２が演算子ｏｐ３として機能し、演算器Ａ３は働かず、記憶素子Ｍ１が演算子ｏｐ１の出力したデータを記憶し、記憶素子Ｍ２が演算子ｏｐ３の出力したデータを記憶する。実行ステップ２において、演算器Ａ１と演算器Ａ２は働かず、演算器Ａ３が演算子ｏｐ２として機能し、記憶素子Ｍ１が演算子ｏｐ２の出力したデータを記憶し、記憶素子Ｍ２は働かない。実行ステップ３において、演算器Ａ１が演算子ｏｐ４として機能し、演算器Ａ２と演算器Ａ３は働かず、記憶素子Ｍ１が演算子ｏｐ４の出力したデータを記憶し、記憶素子Ｍ２は働かない。

【００５６】

図１０の演算子演算器データベースＤ９によれば、演算子ｏｐ１乃至ｏｐ４から、同じ演算子演算器レコード２８内に記憶された演算器Ａ１乃至Ａ３を検索することが可能である。記憶素子Ｍ１、Ｍ２と実行ステップ１乃至３から同じ演算子演算器レコード２８内に記憶された演算子ｏｐ１乃至ｏｐ４を検索することが可能である。

【００５７】

ステップＳ１１において、動作記述Ｄ１のシミュレーションを行う。図１１に示すように、入力データＤ１３として、図４の動作記述Ｄ１の引数ｉ１に２を代入し、引数ｉ２に１を代入した。そして、演算子ｏｐ１からの出力データＤ１０として３を出力した。同様に、演算子ｏｐ２、ｏｐ３、ｏｐ４からの出力データＤ１０として３、１、４を出力した。

【００５８】

次に、動作記述Ｄ１の引数ｉ１に３を代入し、引数ｉ２に２を代入した。そして、演算子ｏｐ１、ｏｐ２、ｏｐ３、ｏｐ４からの出力データＤ１０として５、５、１、６を出力した。さらに、動作記述Ｄ１の引数ｉ１に２を代入し、引数ｉ２に３を代入した。そして、演算子ｏｐ１、ｏｐ２、ｏｐ３、ｏｐ４からの出力データＤ１０として５、－５、－１、－４を出力した。

【００５９】

なお、入力データＤ１３と出力データＤ１０は、図１１に示すように、それぞれフィールドを構成し、入力データＤ１３と出力データＤ１０と演算子フィールド３３からなる入力データ出力データレコード３２を有するデータベースを構成してもよい。入力データ出力データレコード３２を有するデータベースによれば、演算子ｏｐ１乃至ｏｐ４と入力データＤ１３から、同じ入力データ出力データレコード３２内に記憶された出力データＤ１０を検索することが可能である。

【００６０】

ステップＳ５において、ＲＴＬ記述生成部６で、データパスと制御情報Ｄ５に基づいて、ＲＴＬ記述Ｄ６を生成する。ステップＳ６において、ＲＴＬ記述シミュレーション部１１で、実行ステップｉに、最初に実行される実行ステップ１を

設定する。

【００６１】

ステップＳ７において、入力データＤ１３で引数ｉ１が２であり、引数ｉ２が１である場合のRTL記述Ｄ６のシミュレーションを行う。図１２に示すように、まず、実行ステップ１で、RTL記述Ｄ６に入力データＤ１３を代入し、記憶素子Ｍ１、Ｍ２が記憶する記憶データＤ１１を計算する。ＲＴＬ記述Ｄ６の引数ｉ１に２を代入し、引数ｉ２に１を代入した。そして、実行ステップ１で、記憶素子Ｍ１の記憶データＤ１１として３を出力し、記憶素子Ｍ２の記憶データＤ１１として１を出力した。

【００６２】

ステップＳ１２において、図１０の演算子演算器データベースＤ９に基づいて、計算された記憶データＤ１１の実行ステップ１と記憶素子Ｍ１から演算子ｏｐ１を検索する。また、実行ステップ１と記憶素子Ｍ２から演算子ｏｐ３を検索する。

【００６３】

ステップＳ１３において、図１１の検索された演算子ｏｐ１の出力データＤ１０である３と、図１２の実行ステップ１で記憶素子Ｍ１の記憶データＤ１１である３の異同を判定する。３と３で同じである。また、図１１の検索された演算子ｏｐ３の出力データＤ１０である１と、図１２の実行ステップ１で記憶素子Ｍ２の記憶データＤ１１である１の異同を判定する。１と１で同じである。これらにより、ステップＳ１４に進む。

【００６４】

ステップＳ１４において、実行ステップ１が最適化データフロー図Ｄ４の最大の実行ステップ３以上であるか否かを判定する。実行ステップ１が最大の実行ステップ３以上でないので、ステップＳ１５に進む。ステップＳ１５において、実行ステップ１を１つ大きく実行ステップ２にする。そして、ステップＳ７に進む。

【００６５】

再度、ステップＳ７において、入力データＤ１３で引数ｉ１が２であり、引数

ｉ２が１である場合で実行ステップ２のRTL記述Ｄ６のシミュレーションを行う。図１２に示すように、実行ステップ２で、記憶素子Ｍ１の記憶データＤ１１として３を出力した。

【００６６】

ステップＳ１２において、図１０の演算子演算器データベースＤ９に基づいて、実行ステップ２と記憶素子Ｍ１から演算子ｏｐ２を検索する。

【００６７】

ステップＳ１３において、図１１の検索された演算子ｏｐ２の出力データＤ１０である３と、図１２の実行ステップ２で記憶素子Ｍ１の記憶データＤ１１である３の異同を判定する。３と３で同じであるので、ステップＳ１４に進む。

【００６８】

ステップＳ１４において、実行ステップ２が最適化データフロー図Ｄ４の最大の実行ステップ３以上であるか否かを判定する。実行ステップ２が最大の実行ステップ３以上でないので、ステップＳ１５に進む。ステップＳ１５において、実行ステップ２を１つ大きく実行ステップ３にする。そして、ステップＳ７に進む。

【００６９】

三度、ステップＳ７において、入力データＤ１３で引数ｉ１が２であり、引数ｉ２が１である場合で実行ステップ３のRTL記述Ｄ６のシミュレーションを行う。図１２に示すように、実行ステップ３で、記憶素子Ｍ１の記憶データＤ１１として４を出力した。

【００７０】

ステップＳ１２において、図１０の演算子演算器データベースＤ９に基づいて、実行ステップ３と記憶素子Ｍ１から演算子ｏｐ４を検索する。

【００７１】

ステップＳ１３において、図１１の検索された演算子ｏｐ４の出力データＤ１０である４と、図１２の実行ステップ３で記憶素子Ｍ１の記憶データＤ１１である４の異同を判定する。４と４で同じであるので、ステップＳ１４に進む。

【００７２】

ステップＳ１４において、実行ステップ３が最適化データフロー図Ｄ４の最大の実行ステップ３以上であるか否かを判定する。実行ステップ３が最大の実行ステップ３以上であるので、論理回路設計方法をストップする。

【００７３】

このように、動作記述Ｄ１のミュレーション結果である演算子の出力データＤ１０と、RTL記述Ｄ６のシミュレーション結果である記憶素子の記憶データＤ１１が、実行ステップ毎に一致することを確認することで、動作記述からRTL記述が正確に高位合成されたことが検証できた。

【００７４】

（実施例２）

RTL記述Ｄ６のシミュレーションを、1実行ステップ毎にデータを比較をするのでなく、RTL記述Ｄ６の全実行ステップの実行後に動作記述Ｄ１とRTL記述の比較検証を行う場合について説明する。入力データＤ１３は、引数ｉ１が３であり、引数ｉ２が２であるとする。実施例１とステップＳ１乃至Ｓ５、Ｓ８乃至Ｓ１１は同じである。ステップＳ１２、Ｓ１３、Ｓ１６、Ｓ１７の実施に先駆けて、ステップＳ６、Ｓ７、Ｓ１４とＳ１５を実施する。ステップＳ７、Ｓ１４とＳ１５でループを構成する。実行ステップ１から最大の実行ステップ３までの実行ステップｉについて、入力データＤ１３で引数ｉ１が３であり、引数ｉ２が２である場合のRTL記述Ｄ６のシミュレーションを行う。図１２に示すように、まず、実行ステップ１で、記憶素子Ｍ１の記憶データＤ１１として５を出力し、記憶素子Ｍ２の記憶データＤ１１として１を出力した。次に、実行ステップ２で、記憶素子Ｍ１の記憶データＤ１１として５を出力した。最後に、実行ステップ３で、記憶素子Ｍ１の記憶データＤ１１として６を出力した。

【００７５】

ステップＳ１２において、図１０の演算子演算器データベースＤ９に基づいて、計算された記憶データＤ１１の実行ステップ１と記憶素子Ｍ１から演算子ｏｐ１を検索する。また、実行ステップ１と記憶素子Ｍ２から演算子ｏｐ３を検索する。実行ステップ２と記憶素子Ｍ１から演算子ｏｐ２を検索する。実行ステップ３と記憶素子Ｍ１から演算子ｏｐ４を検索する。

【００７６】

ステップＳ１３において、図１１の検索された演算子ｏｐ１の出力データＤ１０である５と、図１２の実行ステップ１で記憶素子Ｍ１の記憶データＤ１１である５の異同を判定する。５と５で同じである。また、図１１の検索された演算子ｏｐ３の出力データＤ１０である１と、図１２の実行ステップ１で記憶素子Ｍ２の記憶データＤ１１である１の異同を判定する。１と１で同じである。図１１の検索された演算子ｏｐ２の出力データＤ１０である５と、図１２の実行ステップ２で記憶素子Ｍ１の記憶データＤ１１である５の異同を判定する。５と５で同じである。図１１の検索された演算子ｏｐ４の出力データＤ１０である６と、図１２の実行ステップ３で記憶素子Ｍ１の記憶データＤ１１である６の異同を判定する。６と６で同じである。これらの判定により、論理回路設計方法をストップする。

【００７７】

このように、動作記述Ｄ１のミュレーション結果である演算子の出力データＤ１０と、RTL記述Ｄ６のシミュレーション結果である記憶素子の記憶データＤ１１を、全実行ステップにわたって一致することを確認することで、動作記述からRTL記述が正確に高位合成されたことが検証できた。

【００７８】

（実施例３）

RTL記述Ｄ６に不具合がある場合について説明する。RTL記述Ｄ６のシミュレーションは、実施例１と同様に、RTL記述Ｄ６の全実行ステップの実行後に動作記述Ｄ１とRTL記述Ｄ６の比較検証を行った。入力データＤ１３は、引数ｉ１が２であり、引数ｉ２が３であるとする。実施例１とステップＳ１乃至Ｓ５、Ｓ８乃至Ｓ１１は同じである。実施例２と同様に、ステップＳ１２、Ｓ１３、Ｓ１６、Ｓ１７の実施に先駆けて、ステップＳ６、Ｓ７、Ｓ１４とＳ１５を実施する。図１２に示すように、まず、実行ステップ１で、記憶素子Ｍ１の記憶データＤ１１として５を出力し、記憶素子Ｍ２の記憶データＤ１１として－１を出力した。次に、実行ステップ２で、記憶素子Ｍ１の記憶データＤ１１として１２７５を出力した。最後に、実行ステップ３で、記憶素子Ｍ１の記憶データＤ１１として１２

７６を出力した。

【００７９】

ステップＳ１２において、図１０の演算子演算器データベースＤ９に基づいて、計算された記憶データＤ１１の実行ステップ１と記憶素子Ｍ１から演算子ｏｐ１を検索する。また、実行ステップ１と記憶素子Ｍ２から演算子ｏｐ３を検索する。実行ステップ２と記憶素子Ｍ１から演算子ｏｐ２を検索する。実行ステップ３と記憶素子Ｍ１から演算子ｏｐ４を検索する。

【００８０】

ステップＳ１３において、図１１の検索された演算子ｏｐ１の出力データＤ１０である５と、図１２の実行ステップ１で記憶素子Ｍ１の記憶データＤ１１である５の異同を判定する。５と５で同じである。また、図１１の検索された演算子ｏｐ３の出力データＤ１０である－１と、図１２の実行ステップ１で記憶素子Ｍ２の記憶データＤ１１である－１の異同を判定する。－１と－１で同じである。図１１の検索された演算子ｏｐ２の出力データＤ１０である－５と、図１２の実行ステップ２で記憶素子Ｍ１の記憶データＤ１１である１２７５の異同を判定する。－５と１２７５は異なる。図１１の検索された演算子ｏｐ４の出力データＤ１０である－４と、図１２の実行ステップ３で記憶素子Ｍ１の記憶データＤ１１である１２７６の異同を判定する。－４と１２７６は異なる。異なる判定が生じたのでステップＳ１６に進む。検索された演算子ｏｐ２とｏｐ４で異なる判定が生じた。

【００８１】

ステップＳ１６において、図１０の演算子演算器データベースＤ９に基づいて、異なる判定が生じた演算子ｏｐ２から演算器Ａ３を検索する。同様に、異なる判定が生じた演算子ｏｐ４から演算器Ａ１を検索する。

【００８２】

ステップＳ１７において、論理回路設計装置１のオペレータが、検索された演算器Ａ３とＡ１に基づいて、ＲＴＬ記述Ｄ６をデバックする。なお、演算器Ａ３は演算器Ａ１より早い実行ステップ２で不具合を発生させている。演算器Ａ３での不具合により、演算器Ａ１の不具合が発生する場合も考えられる。このような

場合は、まず、演算器Ａ３に関してデバックを行えばよい。そして、ステップＳ５に戻る。ステップＳ５に戻ることで、デバックされたＲＴＬ記述Ｄ６の記憶素子の記憶データＤ１１の値が、動作記述Ｄ１の演算子の出力データＤ１０に一致するまで、オペレータはデバックを繰り返すことができる。

【００８３】

このように、動作記述Ｄ１とRTL記述Ｄ６の比較検証をすることにより、不具合が生じた場合には、動作記述Ｄ１とRTL記述Ｄ６の両者での不具合箇所が特定できる。このことにより、RTL記述Ｄ６の修正が容易に行え、設計にかかる時間を大幅に短縮することができる。

【００８４】

（実施例４）

実施例４でも、論理回路設計装置１で論理回路設計方法を用いた。実施例４では最適化を実施した場合について説明する。実施例４の論理回路設計方法は、図３のステップＳ１において、図１３に示すような複数の演算子ｏｐ１乃至ｏｐ３を有する動作記述Ｄ１から、演算子ｏｐ１乃至ｏｐ３を実行する演算ノードＮ１乃至Ｎ３を実行順に配列した未加工データフロー図Ｄ２に変換する。

【００８５】

ステップＳ２において、図１４に示すように、未加工データフロー図Ｄ２に実行ステップ１乃至３を割り当てる。入力データＤ１３の引数ｉ１とｉ２で、演算ノードＮ１において、和算の演算子ｏｐ１を実行する。演算ノードＮ１は、実行ステップ１で実行するという時間情報が添付される。演算ノードＮ１からの出力データを記憶するレジスタＲ１を実行ステップ１の実行後に挿入する。

【００８６】

演算ノードＮ１の演算子ｏｐ１の出力と引数ｉ３で、演算ノードＮ２において、和算の演算子ｏｐ２を実行する。演算ノードＮ２は、実行ステップ２で実行するという時間情報が添付される。演算ノードＮ２からの出力データを記憶するレジスタＲ２を実行ステップ２の実行後に挿入する。

【００８７】

演算ノードＮ２の演算子ｏｐ２の出力と引数ｉ４で、演算ノードＮ３において

、和算の演算子ｏｐ３を実行する。演算ノードＮ３は、実行ステップ３で実行するという時間情報が添付される。演算ノードＮ３からの出力データを記憶するレジスタＲ３を実行ステップ３の実行後に挿入する。

【００８８】

スケジューリング部３は、このように加工した未加工データフロー図Ｄ２を、図１４のスケジューリングデータフロー図Ｄ３として出力する。なお、後述するステップＳ８の理解が容易になるように、ステップＳ８の一部を先行して実施する。図１３の動作記述Ｄ１と図１４のスケジューリングデータフロー図Ｄ３に基づいて、図１５に示すようなノード演算子データベースＤ７を生成する。ノードフィールド２２では、演算ノードＮ１乃至Ｎ３とレジスタＲ１乃至Ｒ３が記憶される。演算子フィールド２３では、演算子ｏｐ１乃至ｏｐ３が記憶される。演算子ｏｐ１乃至ｏｐ３とその演算子ｏｐ１乃至ｏｐ３を実行する演算ノードＮ１乃至Ｎ３が、同じノード演算子レコード２１内に記憶される。また、演算子ｏｐ１乃至ｏｐ３とその演算子ｏｐ１乃至ｏｐ３の出力データを記憶するレジスタＲ１乃至Ｒ３が、同じノード演算子レコード２１内に記憶される。

【００８９】

ステップＳ３において、図１４のスケジューリングデータフロー図Ｄ３を、図１６に示すように最適化し、最適化データフロー図Ｄ４を生成した。直列に処理される和算を並列に処理するように組み替えた。演算ノードN2、N3とレジスタR2が消去されて、代わりに演算ノードN4、N5とレジスタR4が生成された。予想される全実行時間が、実行ステップで３実行ステップから２実行ステップに短縮された。

【００９０】

ステップＳ８において、図１３の動作記述Ｄ１と図１４のスケジューリングデータフロー図Ｄ３と図１６の最適化データフロー図Ｄ４に基づいて、まず、図１５のノード演算子データベースＤ７を生成し、さらに、図１７に示すような、ノード演算子データベースＤ７を生成する。

【００９１】

ノードフィールド２２では、演算ノードＮ１乃至Ｎ５とレジスタＲ１乃至Ｒ４

が記憶される。演算子フィールド２３では、演算子ｏｐ１乃至ｏｐ３が記憶される。演算子ｏｐ１乃至ｏｐ３とその演算子ｏｐ１乃至ｏｐ３を実行する演算ノードＮ１乃至Ｎ４が、同じノード演算子レコード２１内に記憶される。なお、演算子ｏｐ２とｏｐ３に直接対応する演算ノードＮ２とＮ３は、図１６の最適化データフロー図Ｄ４では存在しない。そこで、演算子ｏｐ２とｏｐ３の実行に影響を受ける演算ノードに、演算ノードＮ４とＮ５を設定する。演算子ｏｐ２とｏｐ３とその演算子ｏｐ２とｏｐ３の実行に影響を受ける演算ノードＮ４が、同じノード演算子レコード２１内に記憶される。演算子ｏｐ２とｏｐ３とその演算子ｏｐ２とｏｐ３の実行に影響を受ける演算ノードＮ５が、同じノード演算子レコード２１内に記憶される。

【００９２】

また、演算ノードＮ１と、演算ノードＮ１に対応するレジスタＲ１は、演算子ｏｐ１に関係付けられている。演算ノードＮ４と、演算ノードＮ４に対応するレジスタＲ４は、演算子ｏｐ２、ｏｐ３に関係付けられている。演算ノードＮ５と、演算ノードＮ５に対応するレジスタＲ３は、演算子ｏｐ２、ｏｐ３に関係付けられている。

【００９３】

ステップＳ４において、図１８に示すような論理回路のデータパスＤ５と、このデータパスの制御情報Ｄ５を生成する。データパスＤ５は、演算ノードＮ１、Ｎ５として機能する演算器Ａ１と、演算ノードＮ４として機能する演算器Ａ２と、レジスタＲ１、Ｒ３として機能する記憶素子Ｍ１と、レジスタＲ４として機能する記憶素子Ｍ２と、制御情報Ｄ５に基づいて伝達するデータを切り換えるセレクターＣ１、Ｃ２を有する。

【００９４】

ステップＳ９において、図１６の最適化データフロー図Ｄ４と図１８のデータパスＤ５に基づいて、図１９に示すようなノード演算器データベースＤ８を生成する。図１９のノード演算器データベースＤ８より、実行ステップ１において、演算器Ａ１が演算ノードＮ１として機能し、演算器Ａ２が演算ノードＮ４として機能し、記憶素子Ｍ１がレジスタＲ１として機能し、記憶素子Ｍ２がレジスタＲ

４として機能する。実行ステップ２において、演算器Ａ１が演算ノードＮ５として機能し、記憶素子Ｍ１がレジスタＲ３として機能する。

【００９５】

ステップＳ１０において、図１７のノード演算子データベースＤ７と図１９のノード演算器データベースＤ８に基づいて、図２０に示すような演算子演算器データベースＤ９を生成する。図２０の演算子演算器データベースＤ９を見ると、実行ステップ１において、演算器Ａ１が演算子ｏｐ１として機能し、演算器Ａ２が演算子ｏｐ２、ｏｐ３として機能し、記憶素子Ｍ１が演算子ｏｐ１の出力したデータを記憶し、記憶素子Ｍ２が演算子ｏｐ２、ｏｐ３の出力したデータを記憶する。実行ステップ２において、演算器Ａ１が演算子ｏｐ２、ｏｐ３として機能し、記憶素子Ｍ１が演算子ｏｐ３の出力したデータを記憶する。

【００９６】

図２０の演算子演算器データベースＤ９によれば、演算子ｏｐ１乃至ｏｐ３から、同じ演算子演算器レコード２８内に記憶された演算器Ａ１、Ａ２を検索することが可能である。記憶素子Ｍ１、Ｍ２と実行ステップ１、２から同じ演算子演算器レコード２８内に記憶された演算子ｏｐ１乃至ｏｐ３を検索することが可能である。例えば、記憶素子Ｍ１と実行ステップ１から演算子ｏｐ１を検索できる。そして、演算子ｏｐ１から演算器Ａ１を検索できる。また、記憶素子Ｍ２と実行ステップ１から演算子ｏｐ２、ｏｐ３を検索できる。演算子ｏｐ２、ｏｐ３と実行ステップ１から演算器Ａ２を検索できる。

【００９７】

このように、動作記述Ｄ１からRTL記述Ｄ６を生成する際に、動作記述Ｄ１とRTL記述Ｄ６の両者を対応づける演算子演算器データベースＤ９が生成できる。このことにより、動作記述Ｄ１とRTL記述Ｄ６の比較検証をすることができる。この比較検証で不具合が生じた場合には、動作記述Ｄ１、RTL記述Ｄ６の両者での不具合箇所が特定できる。動作記述Ｄ１やRTL記述Ｄ６の修正が容易に行える。

【００９８】

【発明の効果】

以上説明したように、本発明によれば、動作記述から変換されたＲＴＬ記述で

の論理回路のシミュレーション結果を迅速に解析するための論理回路設計方法を提供できる。

【００９９】

本発明によれば、コンピュータが動作記述から変換されたＲＴＬ記述での論理回路のシミュレーション結果を迅速に解析するための論理回路設計プログラムを提供せきる。

【０１００】

本発明によれば、動作記述から変換されたＲＴＬ記述での論理回路のシミュレーション結果を迅速に解析するための論理回路設計装置を提供できる。

【図面の簡単な説明】

【図１】

本発明の実施の形態に係る論理回路設計装置の構成図である。

【図２】

本発明の実施の形態に係る論理回路設計装置のデータフロー図である。

【図３】

本発明の実施の形態に係る論理回路設計方法のフローチャートである。

【図４】

実施例１の動作記述を表す図である。

【図５】

実施例１の未加工のＤＦＧを表す図である。

【図６】

実施例１のスケジューリング後のＤＦＧを表す図である。

【図７】

実施例１のノード演算子データベースのデータ構造を表す図である。

【図８】

実施例１のデータパスを表す図である。

【図９】

実施例１のノード演算器データベースのデータ構造を表す図である。

【図１０】

実施例１の演算子演算器データベースのデータ構造を表す図である。

【図１１】

実施例１の動作記述のシミュレーションの結果を表す図である。

【図１２】

実施例１のＲＴＬ記述のシミュレーションの結果を表す図である。

【図１３】

実施例４の動作記述を表す図である。

【図１４】

実施例４のスケジューリング後のＤＦＧを表す図である。

【図１５】

実施例４のノード演算子データベースの最適化前のデータ構造を表す図である。

【図１６】

実施例４の最適化後のＤＦＧを表す図である。

【図１７】

実施例４のノード演算子データベースの最適化後のデータ構造を表す図である。

【図１８】

実施例４のデータパスを表す図である。

【図１９】

実施例４のノード演算器データベースのデータ構造を表す図である。

【図２０】

実施例４の演算子演算器データベースのデータ構造を表す図である。

【符号の説明】

１　論理回路設計装置

２　構文解析部

３　スケジューリング部

４　最適化部

５　ハードウェア割付部

６　レジスタ転送レベル（ＲＴＬ）記述生成部

７　ノード演算子対応情報解析部

８　ノード演算器対応情報解析部

９　演算子演算器対応情報解析部

１０　動作記述シミュレーション部

１１　ＲＴＬ記述シミュレーション部

１２　シミュレーション結果比較部

１３　演算子検索部

１４　記憶データ判定部

１５　演算器検索部

１６　入出力部

２１　ノード演算子レコード

２２　ノードフィールド

２３　演算子フィールド

２４　ノード演算器レコード

２５　演算器フィールド

２６　ノードフィールド

２７　実行ステップフィールド

２８　演算子演算器レコード

２９　演算器フィールド

３０　演算子フィールド

３１　実行ステップフィールド

３２　入力データ出力データレコード

３３　演算子フィールド

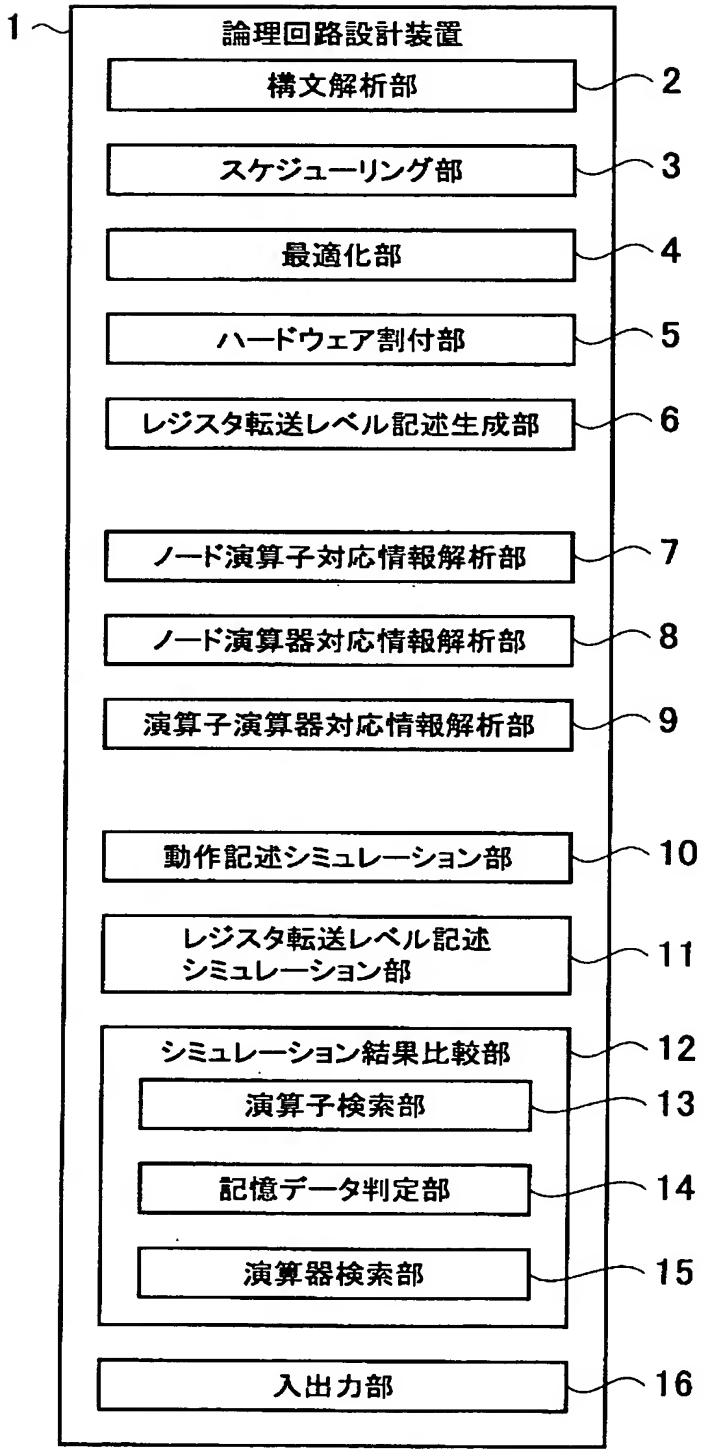
３４　入力データ記憶データレコード

３５　記憶素子フィールド

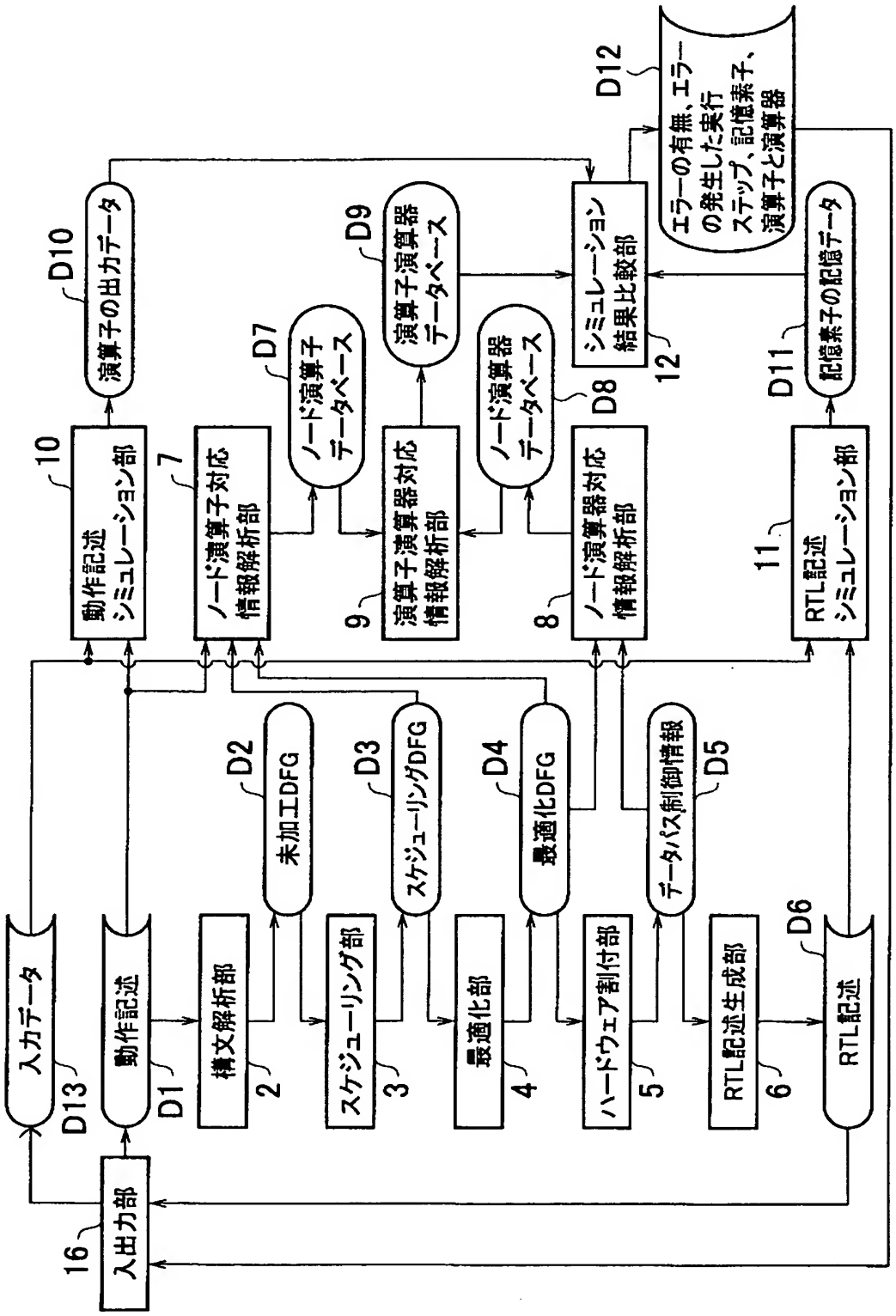
３６　実行ステップフィールド

【書類名】　　図面

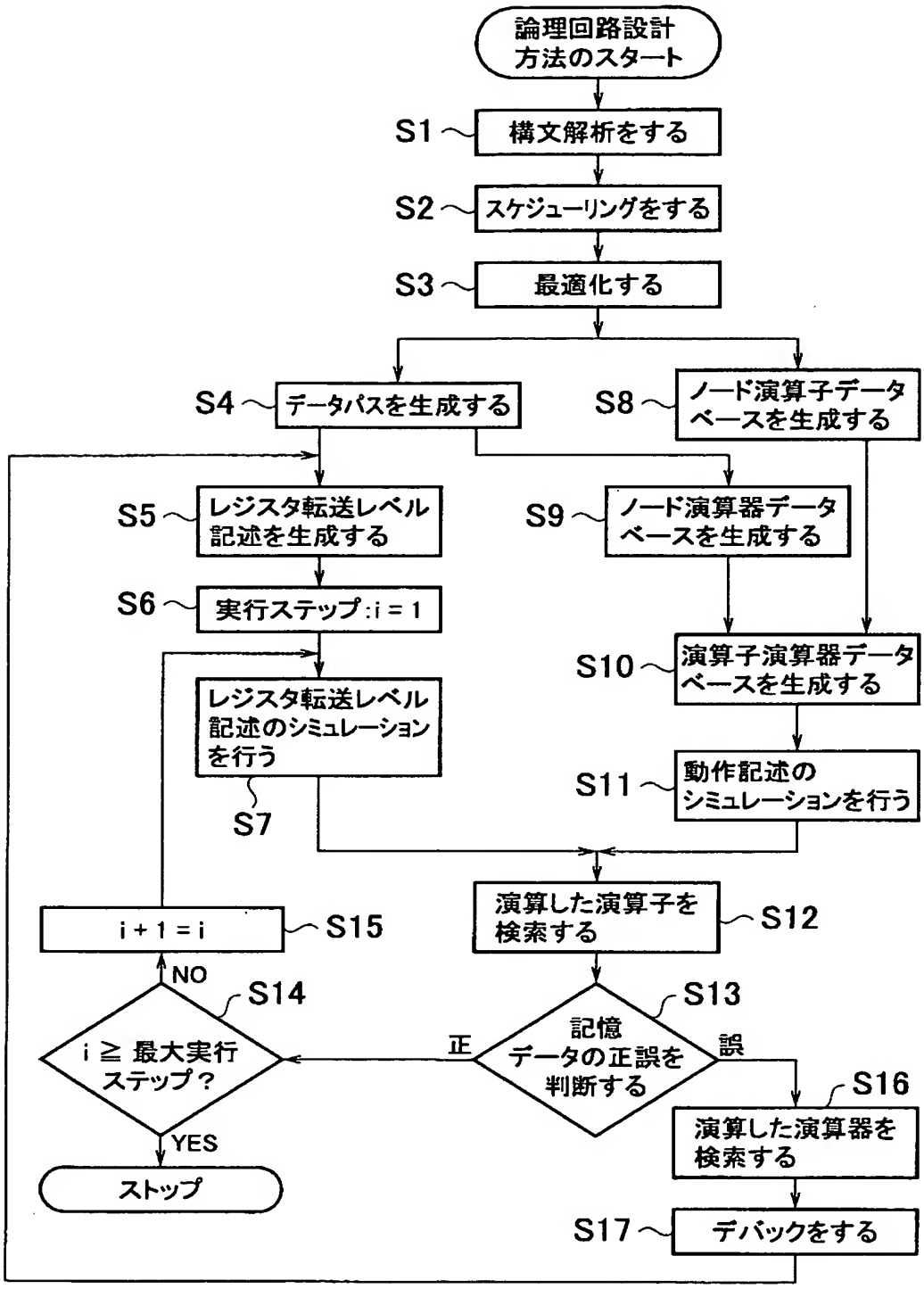
【図１】


1
論理回路設計装置
構文解析部
2
スケジューリング部
3
最適化部
4
ハードウェア割付部
5
レジスタ転送レベル記述生成部
6
ノード演算子対応情報解析部
7
ノード演算器対応情報解析部
8
演算子演算器対応情報解析部
9
動作記述シミュレーション部
10
レジスタ転送レベル記述
シミュレーション部
11
シミュレーション結果比較部
12
演算子検索部
13
記憶データ判定部
14
演算器検索部
15
入出力部
16

【図２】


入力データ
D13
動作記述
D1
構文解析部
2
D2
未加工DFG
スケジューリング部
3
D3
スケジューリングDFG
最適化部
4
D4
最適化DFG
ハードウェア割付部
5
データパス制御情報
D5
RTL記述生成部
6
RTL記述
D6
入出力部
16
動作記述
シミュレーション部
10
演算子の出力データ
D10
ノード演算子対応
情報解析部
7
ノード演算子
データベース
D7
演算子演算器対応
情報解析部
9
演算子演算器
データベース
D9
ノード演算器対応
情報解析部
8
ノード演算器
データベース
D8
シミュレーション
結果比較部
12
エラーの有無、エラーの発生した実行ステップ、記憶素子、演算子と演算器
D12
RTL記述
シミュレーション部
11
記憶素子の記憶データ
D11

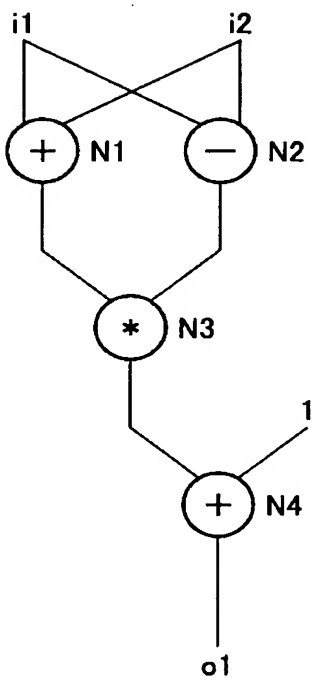
【図３】


論理回路設計
方法のスタート
S1
構文解析をする
S2
スケジューリングをする
S3
最適化する
S4
データパスを生成する
S8
ノード演算子データ
ベースを生成する
S5
レジスタ転送レベル
記述を生成する
S9
ノード演算器データ
ベースを生成する
S6
実行ステップ:i = 1
S10
演算子演算器データ
ベースを生成する
レジスタ転送レベル
記述のシミュレーション
を行う
S7
S11
動作記述の
シミュレーションを行う
演算した演算子を
検索する
S12
i + 1 = i
S15
NO
S14
S13
i ≧ 最大実行
ステップ?
正
記憶
データの正誤を
判断する
誤
S16
演算した演算器を
検索する
YES
ストップ
S17
デバックをする

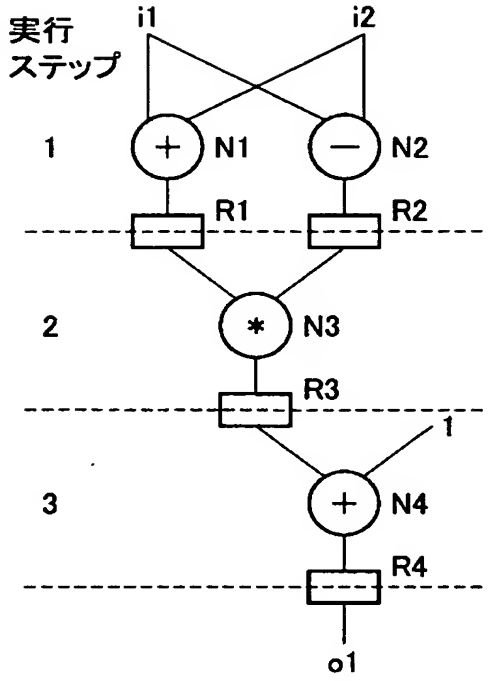
【図４】

```
void calc(int i1, int i2, int &o1)
{
    o1 = (i1 + i2) * (i1 - i2) + 1;
}          op1   op2   op3   op4
```

【図５】



【図６】



【図７】

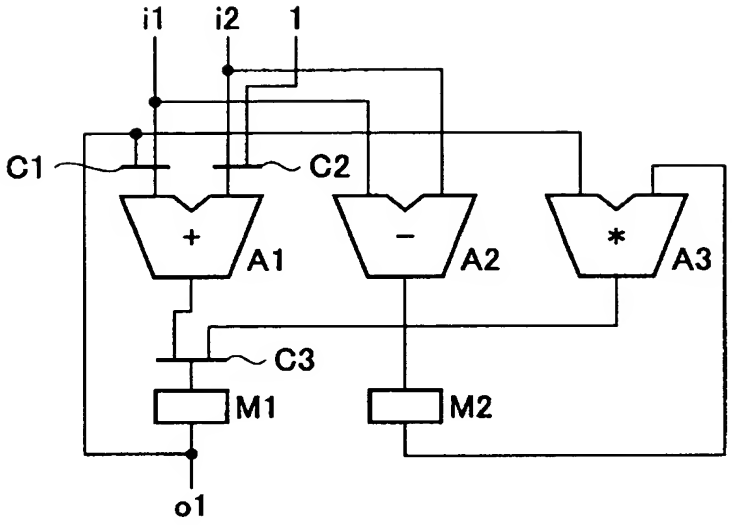
D7

ノード	演算子
N1	op1
N2	op3
N3	op2
N4	op4
R1	op1
R2	op3
R3	op2
R4	op4

21

22　23

【図８】



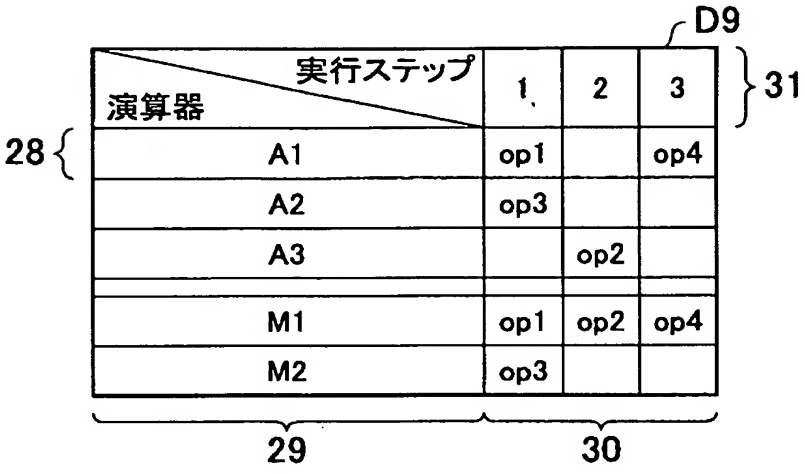
【図９】

D8

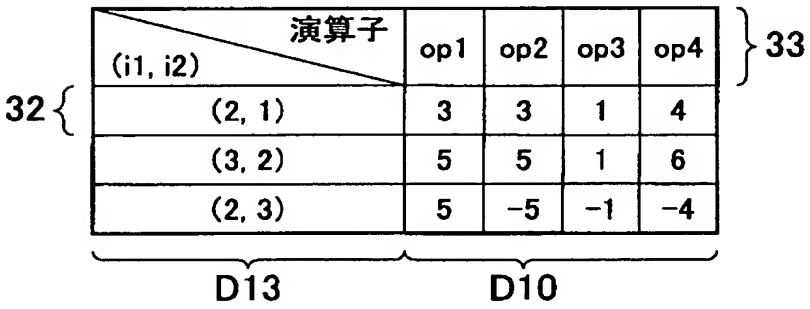
実行ステップ / 演算器	1	2	3
A1	N1		N4
A2	N2		
A3		N3	
M1	R1	R3	R4
M2	R2		

24　27　25　26

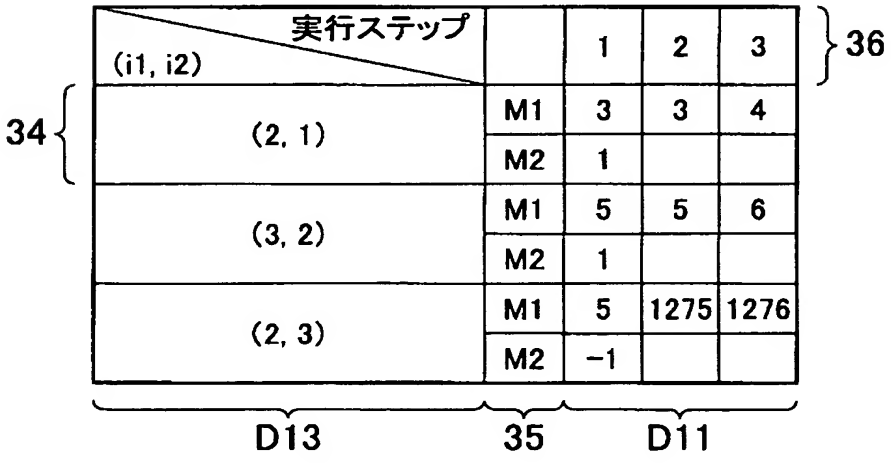
【図１０】


D9
実行ステップ
演算器
1
2
3
31
28
A1
op1
op4
A2
op3
A3
op2
M1
op1
op2
op4
M2
op3
29
30

【図１１】


演算子
(i1, i2)
op1
op2
op3
op4
33
32
(2, 1)
3
3
1
4
(3, 2)
5
5
1
6
(2, 3)
5
−5
−1
−4
D13
D10

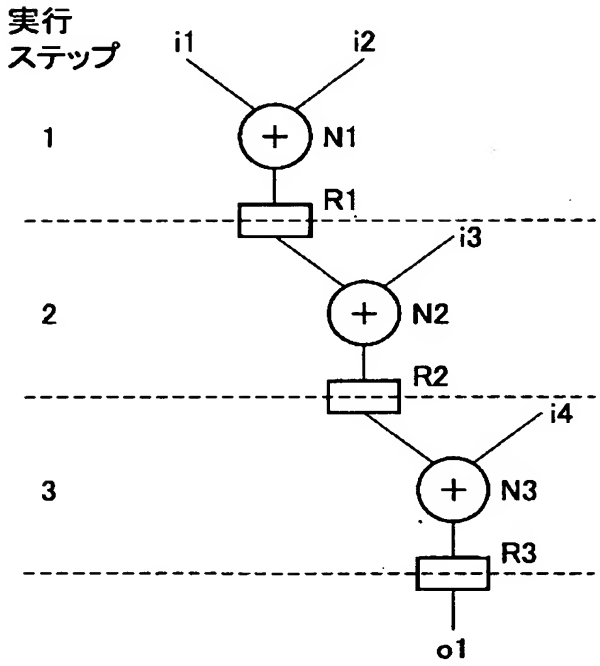
【図１２】


実行ステップ
(i1, i2)
1
2
3
36
34
(2, 1)
M1
3
3
4
M2
1
(3, 2)
M1
5
5
6
M2
1
(2, 3)
M1
5
1275
1276
M2
−1
D13
35
D11

【図１３】

```
void calc(int i1, int i2, int i3, int i4, int &o1)
{
    o1 = i1 + i2 + i3 + i4;
}          op1  op2  op3
```

【図１４】

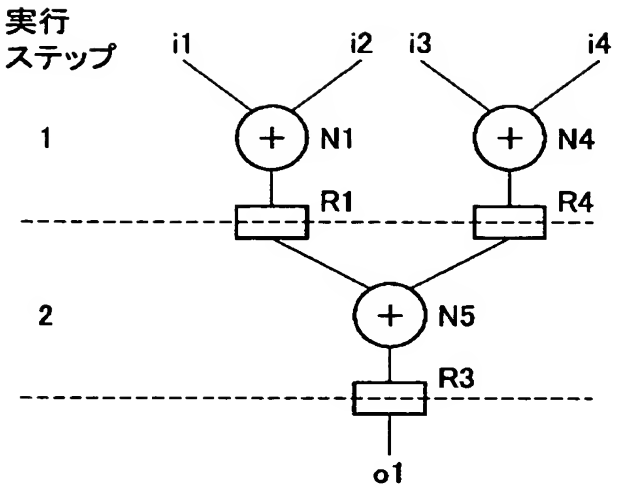


【図１５】

ノード	演算子
N1	op1
N2	op2
N3	op3
R1	op1
R2	op2
R3	op3

21　D7　22　23

【図１６】



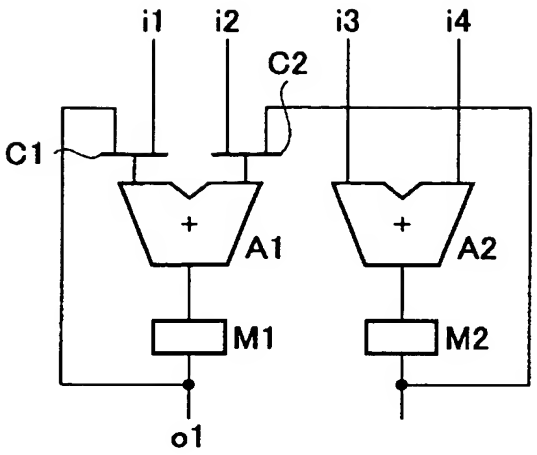
【図１７】

ノード	演算子
N1	op1
N2	op2
N3	op3
N4	op2, op3
N5	op2, op3
R1	op1
R2	op2
R3	op2,op3
R4	op2, op3

D7

22　23

【図１８】



【図１９】

D8

演算器＼実行ステップ	1	2
A1	N1	N5
A2	N4	
M1	R1	R3
M2	R4	

27
24
25
26

【図２０】

D9

演算器＼実行ステップ	1	2
A1	op1	op2, op3
A2	op2, op3	
M1	op1	op2,op3
M2	op2, op3	

31
28
29
30

【書類名】　　要約書

【要約】

【課題】　動作記述から変換されたＲＴＬ記述での論理回路のシミュレーション結果を迅速に解析するための論理回路設計方法を提供する。

【解決手段】　論理回路のアルゴリズムを、複数の演算子を有する動作記述から、演算子を実行する演算ノードを実行順に配列したデータフロー図に変換し、データフロー図に実行ステップを割り当て、演算ノードからの出力データを記憶するレジスタを実行ステップの実行後に挿入する。演算ノードとして機能する演算器とレジスタとして機能する記憶素子を有する論理回路のデータパスと、データパスの制御情報を生成する。演算子を実行する演算器を演算子から検索可能であり、記憶素子の機能をするレジスタが記憶するデータを出力する演算子を実行ステップと記憶素子から検索可能である演算子演算器データベースを生成する。

【選択図】　　図３

出　願　人　履　歴　情　報

識別番号　　〔０００００３０７８〕

１．変更年月日　　２００１年　７月　２日
［変更理由］　住所変更
住　所　　東京都港区芝浦一丁目１番１号
氏　名　　株式会社東芝